Open Source Development Initiative
Strategy and Planning Meeting – June 1, 2012
Minutes

Attendees:

Name	Title, Organization
Deborah Bryant	Director, Open Source Initiative
Andrew Cates	Director, Biomedical Informatics Program, Medical University of South Carolina
Joseph Dal Molin	President, E-cology Corporation; Chairman, WorldVistA
Maria Gaboury	Director, Health Information Technology, LMI
David Hale	Project Manager, Project Pillbox, National Library of Medicine
Luis Ibanez	Technical Leader, Kitware
Jim Jagielski	Director, Open Source Initiative; Co-founder, Apache Foundation
Matt Kennedy	Associate Director for Engineering, NCI
Tony Kerlavage	Program Director, Biomedical Informatics, NCI
Juli Klemm	Associate Director for Life Sciences Programs, NCI
Rob Kolodner, MD	Chief Health Informatics Officer, Open Health Tools
Nina Preuss	Project Manager, TCG
Dave Riley	President and Chairman, Alembic Foundation
John Scott III	Open Technology Lead & Senior Systems Engineer, RadiantBlue Technologies
Eve Shalley	Program Coordinator, Essex Management, NCI
Robert Shirley	Director of Engineering, NCI
Tom Wilson	President, Jack Russell Software

Summary of Key Points:

- **Make it simple, keep barrier to entry as low as possible, and reuse already available assets.**
  - Licenses: NCI should adopt a standard OS license, and move away from the custom license currently in place. The existing licenses for OS are vetted, available, and vetted / well-understood.
  - Code placement and availability:
    - Put the code out there in an easy-to-use, "standard" repository.
    - Don't restrict access through one channel or portal. Provide a way to access, but allow other conduits as well. Use an existing OS repository / community if possible.

- **Don't over-think the process, and stop waiting to do something – fail early!**
  - Don't wait for the code or process to be perfect. Put the applications out there and let them fail or succeed, as determined by the community.
  - Everything should be public and transparent

- **Let the community decide!**
  - The community needs to form around the tools and determine its success or failure.
  - The community needs to determine the appropriate governance processes.
    - NCI can play a role to convene the community, but the decisions themselves need to come from the community itself, not from NCI.
  - NCI will be have a seat at the table, and be a participant, but not have more control than any other community member.
  - Even the concept of "compliance" should be crowd-sourced as much as possible.

- **Purpose-driven projects**
  - Ensure that projects, standards, or "certification" is established based on real-life needs and Use Cases.
  - Don't act as "policemen" to the community.

Detailed Minutes:

Overview

- The attendees introduced themselves and provided some background as to their involvement with Open Source initiatives.

- John Speakman presented the CBIIT project goals.
  - Overview of NCI and CBIIT applications
  - Key points:
    - The government procurement process makes it difficult to deliver effectively on software; "agile" is antithetical to government contracting processes and timeframes.
    - NCI cannot take money or free services, but can take code contributions
    - We want less .gov, more .org – we want to be stakeholders and contribute time and money, but not own the process or control it.

- Juli Klemm provided an overview of what CBIIT has done to-date on the CBIIT OSDI project.
  - CBIIT has begun the process of communicating out to the CBIIT community that we are moving in this direction.
  - Enhancements have been made out in the community, but we have not been able to easily incorporate them into the software and provide them back to the user base.
  - A simple community code resource directory has been established – a web page with tools developed by community members on their own. This does not include the 70+ tools we have developed that we now wish to move to an OS environment.
  - One tool has moved to OS, our caLIMS (Laboratory Information Systems) application. Without much effort on our part, a community has arisen around it since it has been moved to OS.

Topics Discussed:

Structure and Governance

CBIIT is looking for concrete suggestions on moving from the tools being government-driven to community-driven. What structure is appropriate? One idea we had was to set up a non-profit. If we do that, how do we go about it, what are the required roles, and what do we have to do that to ensure it is works effectively?

Responses:

- You have several options, including B-corporation, non-profit, or an LLC where people who join become members. It does not necessarily have to be non-profit, if you structure it for public benefit. However, government probably cannot have an ownership stake in an LLC, so that may not be feasible.

- Should you set up anything new at all? I don't think so. There's enough out there and setting up another new entity will fragment things even more. The right approach is to find an existing community where things can coalesce.

- If you go with an existing community, you need to identify existing problems that will arise from each, and then decide which is most appropriate to meet your needs.

- OS and health systems are bottom-up entities, so you need to design things to encourage self-organization, make it easy for the community to interact and for people to know it's already out there. You can design simplicity into the eco-system. Every time we try to manage natural systems, we mess it up. If we can define the basic elements to drive self-organization and collaboration, then you can decide what minimal structure you need around it.

Testing, Certification, and Trademark

John Speakman explained the caBIG certification process, and how it is included in contracts.

- External groups can apply for "compliance status".
- caBIG has a registered trademark. CBIIT needs to determine if/how to apply that in OS collaborative framework.
- We may need different levels (compliant, accepted, etc.). The old framework has been deprecated for a long time.
- Standards are not always easy to comply with and aren't always a priority for individual research groups. However, it is important to provide a framework that facilitates standards compliance, to keep the barrier as low as possible and create the environment / capability for data sharing. Does the gov't need to be more "controlling" when it comes to standards enforcement?
- To date, we have performed this certification using people, not using a toolkit or automation. We have tried to scale this approach by mentoring and training the community so we can pay them to act as "verifiers". We are leery of continuing with this approach because we are training people to do things that are not part of their natural skills or daily work. What new factors come into play when software is being developed in multiple places?

Responses:

- You need to have a trademark that has value so people are incentivized to go through certification, and the process needs to be easy. If you do have test suite / certification toolkit, it could also be open-sourced, and have the community enhance that.

- This could be a business opportunity for people who want to determine if there is a market for certified versions of these tools. They could bundle it and certify it themselves, or provide a certification service.

- You need to incentivize the community. Hack-a-thons are good. If you have to put cash on the table to get participation, it means they don't believe in the process.

- You should avoid displaying the registered trademark symbol because it is offensive to the community. It tells you it's not yours and you're not invited to participate. You have to be very careful with marketing in OS. You want to create symbols that help people work together.

Are there examples of OSDIs in which testing and certification requirements vary across projects, depending on user requirements? If so, how is this managed?

Responses:

- The OS community responds to the hurdle of certification effectively if it is a thriving community.

- Certification can take on a life of its own. You need to make sure it doesn't become an end in itself. Also, if it's too cumbersome, they won't do it, or it will slow innovation.

- Just list the standard the application is compliant with. Don't get into the trap of "levels of compliance". Don't use a trademark; just indicate with which particular items the tool is compliant.

- Certification criteria should be subject to natural selection, and evidence-based improvement. If there's drift between what the community thinks is important and the certification criteria, they won't do it. The pressure should come from the user community, rather than you being a policeman.

- CDISC could certify as another voice in the community. It's okay to have a separate standards body write the requirements as long as the community can give feedback.

- Instead of forcing the issue, fund real projects that create real demand – utilize a use case to drive the compliance. Otherwise what you are doing is pointless, because no one will use it. Make it purpose-driven, with specific delivery goals.

Can different licenses be applied to separate product components if the code is sufficiently modular? If so, how can the core be protected from being infected with an alternate license?

Responses:

- Use a real, standard license that already exists for OS.
- David Wheeler has a nice matrix on that – the known problems when you combine licenses.
- The objectives are non-viral, no reach thru – in this case there are only a few choices, and Apache2 seems to be the best.
- Send an email to OSI board and ask their opinion.

  Question CBIIT needs to investigate: Is it possible to change the current license?

Tools

What testing tools/frameworks are appropriate to this effort? We currently use SVN for version control. We are moving to a new build approach because the last build utility (BDA) was too cumbersome. Our goal is not to proscribe, but to provide a path to compliance. We want to provide a "starter kit" they can use, or not, and provide some standard of quality.

What SDLC frameworks are useful in OSDI? Should everything be in the same repository?

It seems clear that an automated build and test environment is essential. What about user acceptance testing? How can this be coordinated/accomplished?

Responses:

- Recommendation for a testing framework: "Cucumber" (http://cukes.info/).
- Perhaps certify developers instead of software. This would align with the three reasons for doing OS: autonomy, mastery, purpose.
- This is related to governance, so the community should decide. Some people are paid by companies to participate, because they get benefit, so there are different reasons for participating.
- Educate them, show them multiple models, and then let them pick a governance approach.
- Ask the community, they can decide, you can show them examples. It may be how someone can discover the capabilities he/she is seeking (*e.g.,* through a portal) that is important versus where the assets are physically stored.
- We make testing part of the release cycle, which was decided by the community. If your community is engaged they will make you aware if something is broken. The community

process ensures certain level of quality.

- If you do provide testing tools, they need to be simple, automated, and always available.
- Participants put the effort into the software and participating in the community on behalf of someone, so they are motivated to make sure it works, because they need it.

Community Engagement – Decision-making

What issues do we ask the community to decide upon, and which should CBIIT decide?

Responses:

- First educate the community, then:
  - Organization – CBIIT should decide the legal structure (non-profit, etc.).
  - Governance should be decided by the community. How does governance evolve? Again, ask the community. There needs to be a structure for how decisions are made, how community will collectively make the decisions. Ask the community how they see themselves succeeding in this "meritocratic" world.
  - Remove the word "custodial" from your vocabulary, and flip the question to clarify how you see your involvement evolving over time. You have to rid yourself of the idea of the influence of small versus large organizations. In an OS environment, you don't have corporations; you have individuals who may be participating on behalf of a corporation, large or small. All participants are the same – each one participates on their own. Your sphere of influence grows based on what you do in the community. That's part of the meritocracy.

Community-building and Participation

What are the considerations for establishing trust and collaboration across a heterogeneous community (*e.g.*, vendors, academic, government)? How do you get people to commit to participating?

Our mission is different from a commercial entity, in that our mission is to understand cancer. To that end, we want to steward access to digital capabilities (data and tools) that are essential for researchers. Our community is the cancer research community.

Community participation will need to be motivated by merit and by benefit to their research, rather than by direct funding.

Responses:

- Go to colleges, train them. Incubators *(e.g.* mHealth) and sponsored programs like Google Summer of Code are also good ways to build community. Bring the mission to the community so it is meaningful to them - create evangelists.

- Perhaps you can offer a grant to do the education in the community.

- Ask the community itself to help figure out what the new role for the community should be.

- You need to be careful - fast growth is not always what you want. You need to be sure you have users. Find early adopters / sponsors, let it start growing and make the transition, allow it take hold and have solid footing.

- Make sure to acknowledge the developers, even for small contributions. When they see their name on the developers list for a release, this is a big motivator.

- PIs may be motivated to engage, not just for the benefit they'll receive from the software, but also to strengthen the relationship with NCI.

- We talk about community but we are really talking about developers first. You need to be where the developers hang out, if they are not on-board, it won't work.

- We've seen in healthcare that the most successful places are where the developers and other stakeholders are working together. What you're describing works in more technical areas, but not in life sciences, because both have to be comfortable. For OS to succeed you need that tight feedback loop between those groups. You need to tap into that passion of someone who needs to "scratch his/her own itch" – something that drives development.

- Continue the talk-shop aspect of your community facilitation. Ensure the communications and expectations are clear, and be consistent in how you present your role, as another entity at the table.

    - It's the community, not the tools. It's the governance, not the technology.

    - NCI may be the stable and consistent entity that reminds the community why they are doing this. You can convene, but not control – facilitate the discussion, then back off.

Process

What processes have you established that have been helpful in managing the community and the quality of the code? We would like a robust process, and we also have to be concerned with 508-compliance if an instance will be hosted at NCI. Even though most of the tools are not intended for government use, we may need to consider the government standards.

Responses:

- We have a strict definition of potentially "shippable code", and our goal is to have it every day. Everything submitted goes through a smoke test, it either passes or gets backed out. If there are new capabilities that the community wants, the community verifies and tests that they have been implemented correctly. We then deploy the code to a deployment server, do code analysis (for defects and style consistency) and then run install procedures. This may seem onerous at first, but after you get the rhythm, you know it works, and you have something new

every day.

- We let the community decide what's in beta, or ready for production. We also have ratings and reviews that are community-hosted and written.

Transition

There is great variability among all these tools. Firstly, we have everything from small widgets to huge infrastructure. Second, the degree of usage/uptake is very variable, from widely-deployed to almost no uptake at all. Third, some tools have great utility, and others are more limited. Some may be useful but buggy; others are more stable but are lacking a couple of features. We don't want throw everything out and have it become abandon-ware. How do we coalesce communities of interest around these tools? We have 70 things, many of which are unrelated to each other. Some things will be abandoned, and it won't be worth our effort, so we will need to be judicious with our time and money.

At the same time, we have been surprised by what happened when we moved the caLIMS product to Open Source. Since we have put it out there, a community has arisen and self-organized. This means we may not know which will succeed, and which will not, and we have to subject each application to the tyranny of the market.

Responses:

- Rather than you taking the action, the community can self-organize. Start with the capabilities that have the strongest community, learn from those, and take the processes that help the community form. Rather than you driving that, you let the community.
- As more people use those tools, they will see the pieces of functionality that are usable in other environments. That will happen organically once awareness is there.

Transition

What are the considerations for code readiness, modularity, and documentation that will be required for successful transitioning projects from CBIIT-sponsored development to mixed development?

Responses:

- Create the software with a package manager; the more you can have people contribute without dealing with the core, the better.
- Bring in the old programmers to train the community. Ask them to modularize the components
- At a minimum create good documentation. You want knowledge transfer so participants can take over the project. Real open source developers are going to want documentation and a way to know if a bug is fixed. They'll need something that provides high-level architectural viewpoint

on how code flows, APIs and dependencies, functional aspects and sections of the code (where to find functionality), and how to deploy it.

- Provide a sandbox for the first year.

Metrics for Success

What are useful success metrics that have been used by other projects that would be relevant for the caBIG OSDI?

Responses:

- Provide lots of transparency - who funded what, how many hits the application receives, what has been done, code commits, and so on, so people can see what's alive and well and what isn't.
- If you can track how many different developers are involved, and you have a diverse group, that's a good sign. It's preferable to have a group participating, rather than one person doing all the work.

- Read the discussion boards and see how many questions are asked and answered.

- Perhaps make sure the infrastructure provides statistics and a real-time "health score".